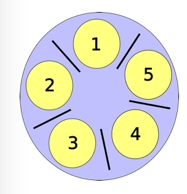
CS241 Lecture 20 Dining Philosophers

typedef struct p {

pthread\_mutex\_t \*fork\_lft, \*fork\_rgt;

const char \*name;

 pthread\_t thread;

int fail;

} Philosopher;

int running = 1;

int main()

{

const char \*nameList[] = { "Kant", "Guatma", "Russel", "Aristotle", "Bart" };

pthread\_mutex\_t forks[5];

Philosopher philosophers[5];

Philosopher \*phil;

int i;

int failed;

for (i=0;i<5; i++) {

failed = pthread\_mutex\_init(&forks[i], NULL);

if (failed) {

printf("Failed to initialize mutexes.");

exit(1);

}

}

for (i=0;i<5; i++) {

phil = &philosophers[i];

phil->name = nameList[i];

phil->fork\_lft = &forks[i];

phil->fork\_rgt = &forks[(i+1)%5];

phil->fail = pthread\_create( &phil->thread, NULL, PhilPhunction, phil);

}

sleep(40);

running = 0;

printf("cleanup time\n");

for(i=0; i<5; i++) {

phil = &philosophers[i];

if ( !phil->fail && pthread\_join( phil->thread, NULL) ) {

printf("error joining thread for %s", phil->name);

exit(1);

}

}

return 0;

}

//http://rosettacode.org/wiki/Dining\_philosophers#C

void \*PhilPhunction(void \*p) {

Philosopher \*phil = (Philosopher\*)p;

int failed;

int tries\_left;

pthread\_mutex\_t \*fork\_lft, \*fork\_rgt, \*fork\_tmp;

while (running) {

printf("%s is sleeping --er thinking\n", phil->name);

sleep( 1+ rand()%8);

fork\_lft = phil->fork\_lft;

fork\_rgt = phil->fork\_rgt;

printf("%s is hungry\n", phil->name);

tries\_left = 2; /\* try twice before being forceful \*/

do {

pthread\_mutex\_lock( fork\_lft);

failed = (tries\_left>0)? pthread\_mutex\_trylock( fork\_rgt )

: pthread\_mutex\_lock(fork\_rgt);

if (failed) {

pthread\_mutex\_unlock( fork\_lft);

fork\_tmp = fork\_lft;

fork\_lft = fork\_rgt;

fork\_rgt = fork\_tmp;

tries\_left -= 1;

}

} while(failed && running);

if (!failed) {

printf("%s is eating\n", phil->name);

sleep( 1+ rand() % 8);

pthread\_mutex\_unlock( fork\_rgt);

pthread\_mutex\_unlock( fork\_lft);

}

}

return NULL;

}